Francis L. Schneider NASA Detailee
Jet Propulsion Laboratory/
NASA IVV Facility
100 University Drive, Fairmont WV 26554
sch@atlantis.ivv.nasa.gov
304-367-8304, 304-367-8211(FAX)
and

John R. Callahan
NASA&WVU Software Research Lab
NASA IV&V Facility
100 University Drive, Fairmont WV 26554
callahan@atlantis.ivv.nasa.gov
304-367-8235, 304-267-8211(FAX)
and

Steve Easterbrook
NASA/WVU Software Research Lab
NASA IV&V Facility
100 University Drive, Fairmont WV 26554
steve@atlantis.ivv.nasa.gov
304-367-8352, 304-267-8211(FAX)
20 May 1997

A Fault Analysis Case Study of the Cassisni CDS Mark and Rollback Design

July 3, 1997

1 Model and Case 1

This white paper summarizes the key findings from a formal analysis of the Cassini CDS Mark and Rollback design, as described in Cassini Design Note 10. The paper describes three anomalies in the design, assesses the risk involved, and makes recommendations for further work to determine whether these errors affect the as-built system. Details of the formal analysis technique are omitted here for brevity; these will be described in a separate paper.

The work described in this paper is a continuation of the analysis of the Mark and Rollback scheme conducted under task 2 of the MOA between the IVV Facility and JPL. The continuation work was funded as a Code Q RTOP research project, as an exploration the use of the formal modelling technique.

We constructed a software validation model of the CDS mark and rollback process, based on the description in Design Note 10 [1]. The SPIN/PROMELA modeling system (See [2]) was used to construct the computer model and to run validation tests on the model. Six different fault categories were identified to test the model. The results reported here cover the first of these categories only, but we do discuss implications for the other five fault categories. Fault category 1 refers to the behavior of the CDS prime string in the face of a peripheral interfering fault.

Six separate requirements on the rollback scheme were validated. These were:

1. rollback required in prime and online to 6BEGIN_PROG when up to three seconds and more than three seconds had gone by (2 requirements);
2. rollback required in prime and online to next previous mark point if up to 3 seconds have gone by (2 requirements) and execution point not at 6BEGIN_PROG;
3. rollback to current mark point in prime and online strings if more than 3 seconds have gone by and not at 6BEGIN_PROG.

To accomplish this, a simple but general critical sequence was constructed and executed using the model. Each of the 6 requirements involved exhaustive examination of approximately 100,000 states in the model, and took about 30 seconds. The response and recovery in each case was to the injection of a single peripheral interfering fault in all possible ways based on the model. Three of the 6 runs for the 6 requirements failed in the verification.

2 Findings

Three anomalies were identified and are described below. The first two are potential errors in the design note that we suspect will not occur in the CDS implementation. The third, we believe, is a discrepancy in the design requirements that could allow for erroneous behavior of the implemented system.

All of the behaviors described used spatial synchronization only. That is, both the prime and the online strings were synchronized initially by forcing them to begin execution using a program counter value of 1 (= 6BEGIN_PROG) and thereafter used a rendezvous handshake during interrupt 5 to maintain synchronization between the prime and online system.

2.1 Anomaly One

Depending on how error detection and repair is handled, it may be possible for the prime system to detect and to repair an intermitent error within one second, and then consequently not to broadcast this state to the online system. This would mean that the online sytem would not receive notice of the fault; therefore, it would continue executing its copy of the critical sequence. Repeated occurrence of this scenario would cause the online system to get way ahead of the prime string, possibly to the point where the online string would complete execution of its copy of the sequence. If the prime string subsequently fails, the online string may not have a markpoint to roll back to.

Details are as follows. The synchronization between prime and online is based on the state of the prime string at the one second boundaries. The STB handshake in RTI_5 will not report the occurrence of a fault that occurred after the beginning of the one second time frame. However, if the fault is repaired before the start of next one-second time frame, the prime string will not report it in the STB at the next handshake either. The result is that the online system is now one second ahead of the prime string. Figure 1 shows one sequence of events in this scenario. Inspection of the figure shows that repeated injection of this type of intermitent fault effectively stalls the prime execution while the online system will eventually complete executing its copy of the critical sequence.

This anomaly is due entirely to the ordering of processing described in the design note. It is possible that the anomaly is not present in the implementation. If it does occur in the implementation, a simple fix can be made by arranging the order of processing somewhat differently.

2.2 Anomaly Two

This anomaly depends upon how faults are handled at the end of a critical sequence. If a fault occurs in the prime string within two seconds *after* the end of the critical sequence is reached, it is not clear how the rollback if any would be handled. Design Note 10 does not designate the 6END_PROG instruction as a mark point. Figure 2 shows this scenario. Our validation run failed because our model assumed that once the critical sequence completed, the prime system returned to the Power Up Idle state; accordingly there would be no suspended critical sequence to return to once the fault was corrected. If the fault were to bring the prime system down, the online system may need to roll back to the last aged markpoint. Presumably then a similar problem would develop here too.

This anomaly is due to a missing requirement. It is likely that the problem does not occur in the implementation. If the problem does occur, a simple fix would be to always treat the END_PROG instruction as a markpoint, and to ensure the mark and rollback process continues until this markpoint is fully aged.

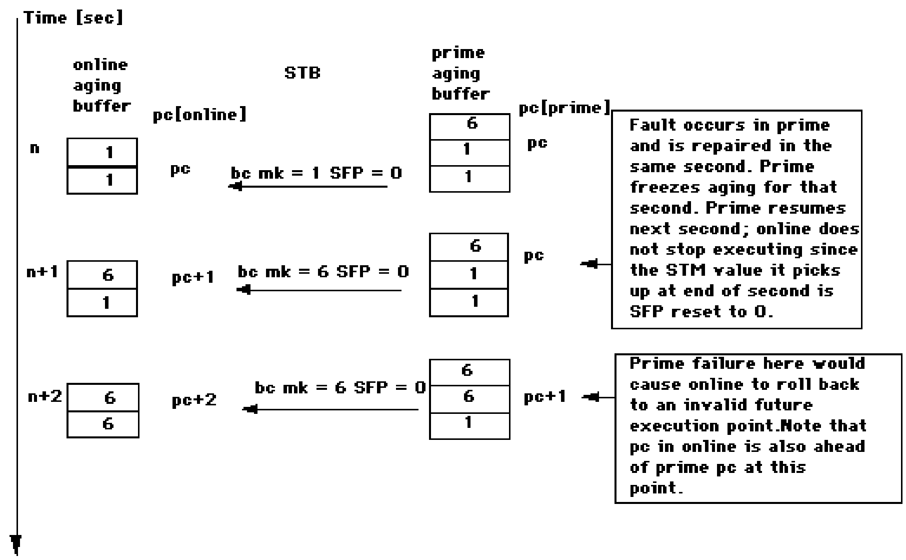

Figure 1: GDRS Prime Fault Repaired in One Second

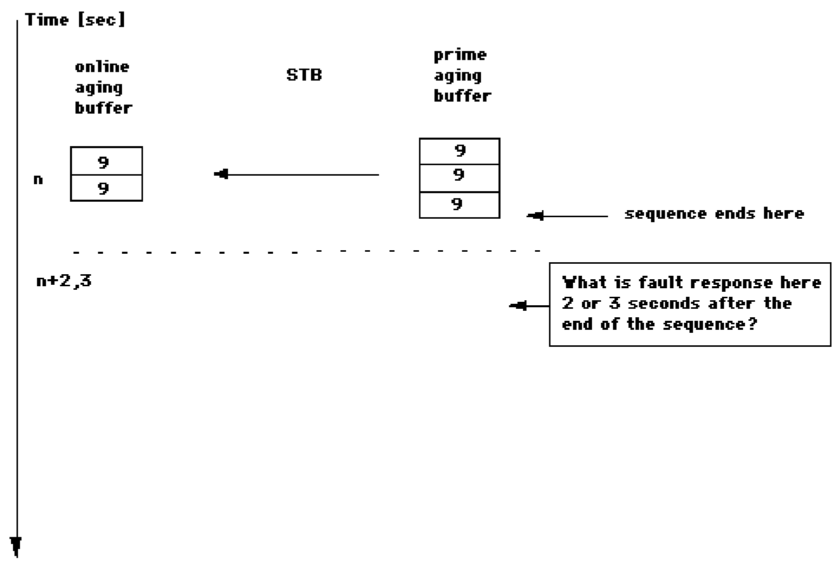

Figure 2: GDRS Fault Two or Three Seconds After Sequence Completion

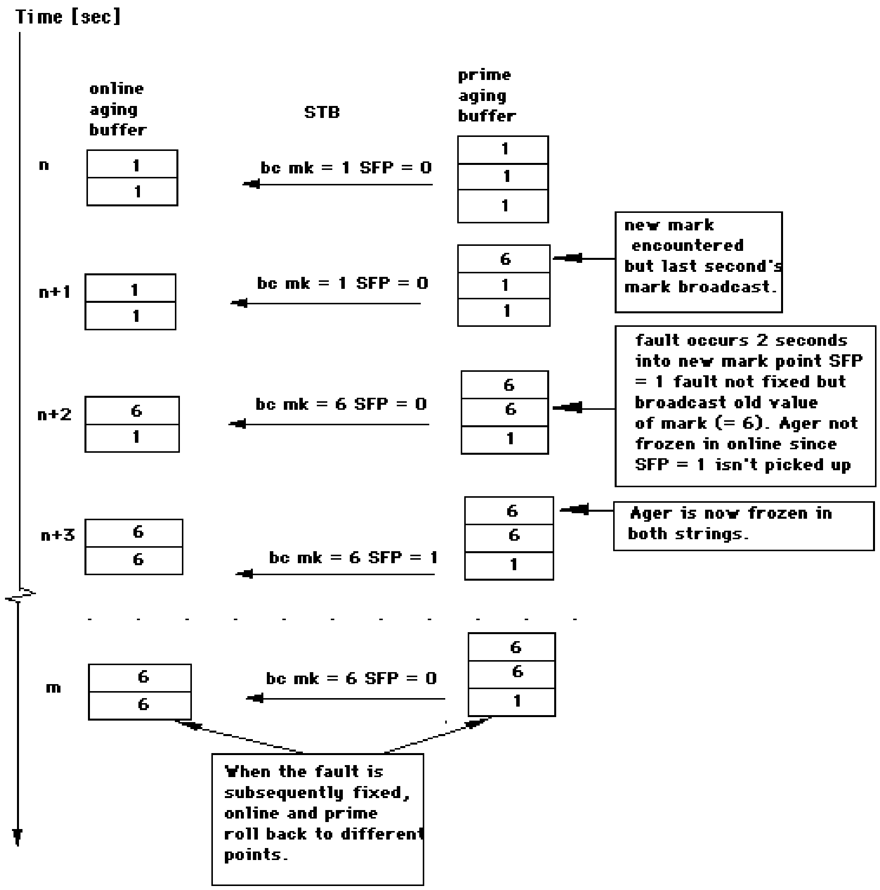

Figure 3: GDRS Prime Fault Two Seconds After Mark Point

2.3 Anomaly 3

This anomaly concerns the occurrence of a fault 2 seconds after a mark point is encountered in the prime string. The situation is shown in Figure 3. The prime system freezes the aging function giving the aging buffer snapshot shown on the right of the figure at n + 2 seconds. Since the STB handshake uses the previous second's value for the flags, SFP = 0 is broadcast. The online string continues to execute and ages its mark point by one further second, giving the configuration shown on the left in the figure at n + 3 seconds. At this point the online system receives the SFP = 1 value and now both agers are frozen. When the fault is subsequently repaired, the prime system will roll back correctly, but the online system will roll to a different point as shown at second m on the left in the figure. This would not cause a problem if the prime system completes the critical sequence. However, if the online system should subsequently have to take over due to a prime failure - possibly associated with the [symptomatic] peripheral interfering fault that was just processed, it could roll to an inappropriate block of code.

Using a comparison of the internal and the external (broadcast) agers in the online system would not seem to help since the broadcast ager is deemed to be the more reliable source. Also, this problem would not go away if the aging buffers were made deeper or shallower. It would just occur at a different place since it is a consequence of the relative time difference between the two aging schemes.

3 summary and recommendations

The analysis technique used in this study is new, and was not sufficiently mature just a few years ago to enable its use. The most thorny problems in systems development occur where communication of vital information is a key requirement. With respect to the mark and rollback process, the CDS operates as a communication system [3]; and in this respect has all of the complexity associated with the such systems. The use of model checkers opens up new possibilities for validating such systems, as illustrated by a statement made by Holzmann [2] in reference to a couple of relatively simple communciation protocols:

> It is almost impossible to manually verify correctness requirements such as the ones discussed, no matter how diligent or

> disciplined the designer. The behavior of even simple protocol systems can be of a complexity that no designer can be expected to assess accurately.

The findings reported here are based upon an analysis of the preliminary version of Design note 10 shared with us by the project; which is now two years old. Hence, it is not known whether the anomalies we have described have already been addressed in revisions of the design note, or in the implementation itself. In either case, it should be fairly easy to determine whether the implementation exhibits these anomalies through inspection and/or testing.

Recommendation 1: Investigate whether the anomalies described in this white paper occur in the implementation of the CDS mark and rollback system. Test cases that exercise these anomalies can be derived from the traces output by the model checker used in this study.

Recommendation 2: The analysis that revealed the anomalies in this area needs to be completed. Firstly, the model can be updated to reflect design changes that have occurred since the preliminary version of Design note 10, from which we have been working. The remaining error cases can then be explored, as an additional validation of the mark and rollback design

Recommendation 3: The model developed in this work can be used to test the coverage of the test suite used for CDS. A simple way of doing this is to seed errors into the model, generate traces through the model that then violate the requirements, and use these traces as test cases for the implementation.

References

[1] A. Elson. Design note 10. Technical report, Jet Propulsion Laboratory, California Institute of Technology, January 1996.

[2] G.J. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall, 1991.

[3] F. L. Schneider. Task2 criticalsequence analysis. JPL Internal Report,Jet Propulsion Laboratory, Pasadena, California, 1997.